LCPROGRAM EXECUTION: A DEMONSTRATION OF COUNT METHOD IN MARRIAGE PROBLEMS.

FRANK K. APPIAH[1]
UNIVERSITY OF CAMBRIDGE
KING'S COLLEGE
CAMBRIDGE, UK.

TEACHING APPOINTMENT

(Applied Mathematics and Computer Science)

1 Dr. Frank Appiah hold studentships at the Bsc. (Bachelor in computer engineering, Kwame Nkrumah University of Science and Technology) in 2008, Msc.(software engineering, King's College London) in 2009/10 and 2 doctorates of philosophy (DPhil / PhD) in computer engineering from Kwame Nkrumah University of Science and Technology in 2013/14. He did his doctor of philosophy in computer science and engineering at King's College London in 2010/12 and postgraduate studies at the same place. He is a member of King's Alumni Group of Engineering, British Academy and Royal Society.

This is a teaching appointment in its first publication at the above institution. Any omitting or errors are mine only and can be communicated to when needed.

ABSTRACT

Letter combinatorics is about sentences or phrases and counting problems or methods. It is a logical structure in a computer code developed by a Java programming language and involves discrete computations like subtraction, addition and multiplication. This demonstrates the program solution of combinatorics of sentences or phrases or words by a Letter Combinatorics (*LC*)[4, 5] program. A Marriage Problem (*MP*) solving is made up of running LCMain, LCProgram and its classes.

1-0 INTRODUCTION

I will describe the input methodology of the LCProgram. Input methodology is about the getting data into the program during execution or running. A program needs data to be able to operate. The process of of placing values from outside data set into varibales in a program is called input. Keyboard is an input device that gives data to a program. File or auxiliary input devices can also do this.

1-1 READ

Java[3] gives us a standard method for inputting data called the *read statement*. Its name is descriptive of its function. The computer reads what you type into the input stream called *static input stream.* It can be found in the static *java.lang.System.in*. Here, is the syntax

template for a read statement:

```
int read(byte b[]) throws IOException
int read() throws IOException
int read(byte b[], int off, int len)
```

```
long skip(long n)
int available()
void close()
void mark(int readlimit)
```

Reads some number of bytes from the input stream and stores them into the buffer array, b. The number of bytes actually read is returned as an integer. This method blocks until input data is available, end of file is detected, or an exception is thrown.

InputStream methods	*Description*
`read(byte b[])`	Reads some number of bytes from the input stream and stores them into the buffer array, b.

InputStream methods	*Description*
`read()`	Reads the next byte of data from the input stream. The value byte is returned as an intin the range 0 to 255.
`read(byte b[], int off, int len)`	Reads up to len bytes of data from the input stream into an array of bytes. An attempt is made to read as many as len bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.
`skip(long n)`	Skips over and discards , n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly, 0.
`available()`	Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream.
`close()`	Closes this input stream and releases any system resources associated with the stream.
`mark(int readlimit)`	Marks the current position in this input stream. A sub-sequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

1-1-1 TRICKS ON READ CODE

Byte Initialization

A *read* method will require an initialized byte before calling can be done.

```
byte data=new byte[n] , where n > 1 to
INTEGER.MAXVALUE
```

The number of letter and numerics is shown as below:

Byte Size	*Number of Alphabets or Alphanumerics*
1	1
2	2
4	4
80	8
1000	1000

For integer reading, it is quite different here. The byte size is the number of digits given by the formula, 10^n to 10^n -1.

Byte Size	*Number of Digits*
1	0-9
2	10-99
3	100-999
4	10000-99999
5	100000- 99999

The byte size for LCProgram is the highlighted row value.

Read-Eval-Loop Code

```
while (true) {
                out.println("\nSelect an item
            from menu(00: Menu
    Items)>>>");
                int value = 1;
```

```
            try {
                int v = in.read(menu);
                String s = new
String(menu);
                value =Integer.valueOf(s);
            }
        catch (NumberFormatException e) {
                int v = in.read(menu);
                String s = new
String(menu);
                value =
Integer.valueOf(s);
            }
        }
```

This is the complete read-eval-loop for the LCProgram.

MultiRead Code with For statement

Due to system error in reading, code has to repeated in the try-catch block code. This is case for most reading. For example, trying to read 5 inputs can leave you with only 4 inputs instead. For statement is normally used.

Here is the trick, for a 5 data input prompt, you have to use for-initialization value of 6 because no matter what one input error will be cause. With the system error, you are still left with the require inputs desired.

```
for(int i=0;i<6;i++)
{
try{ //read code }
catch(Exception e){//same read code}
}
```

1-2 PRINT

Java[3] supports printing from the filter output stream. A *PrintStream* adds functionality to another output stream, namely the ability to print representations of various data values conveniently. Here, is the syntax template for a *print* statement:

void print(char s[]);	void println(char s[]);
void print(double d);	void println(double d);
void print(int i);	void println(int i);
void print(long l);	void println(long l);
void print(float f);	void println(float f);
void print(String s);	void println(String s);
void print(Object o);	void println(Object o);

PrintStream methods and description.

PrintStream methods	*Description*
void print(char c)	Prints an character.
void print(int i)	Prints an integer.
void print(long l)	Prints an long.
void print(float f)	Prints an float.
void print(double d)	Prints an double.
void print(char s[])	Prints an character array.
void println(char c)	Prints a character and then terminate the line.

PrintStream methods	*Description*
void println(int i)	Prints a integer and then terminate the line.
void println(long l)	Prints a long and then terminate the line.
void println(double d)	Prints a double and then terminate the line.
void println(String s)	Prints a string and then terminate the line.
boolean println(Object o)	Prints a object and then terminate the line.

2-0 INTERACTIVE INPUT AND OUTPUT

In order to get input to the program, input prompts are used along a printed message to explain what the user should enter[2].

```
User Prompt Message: Select an item from menu(00: Menu Items)>>>
```

```
Select an item from menu(00: Menu Items)>>>00
```

A *00* input data shows the LCProgram Menu details as below:

```
LCProgram Menu
Explicit Count Methods
==========================

01:     Problem Size
02:     MP Sentences
```

```
03:    Principle Values (MPSet)
04:    Alpha Sizes
05:    Partition Sets
06:    Count Problems
07:    Alpha Label Tokens
08:    MP Sets
09:    Principle Values (RMPSet)
10:    Alphanumerics
11:    Meta-Operation
12:    Principle Values (LMPSet)
13:    File MP Reader
14:    Change Problem Size
15:    Change LC Problems[Interactive]
16:    LCTable View
17:    Partition of Integers
18:    Equality Principles
19:    Show Ferrers Diagrams
20:    Permutation and Combination
21:    One-to-One Correspondence
22:    Possible Number of Orders
23:    Show Pascal Triangle
Press Ctrl+C: Quit LCProgram
============================
```

Letter Combinatorics(LC) has the following program requirements:

1. There is a size for problem sentences equal to 5.

2. A countable number of sentences or phrases.
3. Counting the size of selected phrases for likely occurrence of subset equality of letters is called *count*.
4. A sentence with the number of letters specified is called Π.
5. The logical structure of arithmetic such as +, - and = should be applied.
6. Discrete operations must include count, addition, subtraction and sizes.
7. The sizes of selected phrases are enumerated.
8. Proofs with the discrete operations on which the enumeration of sizes stops.
9. There is a possibility of summation equal to Π.

The LC[4, 5] requirements are used to develop the problem solving cases of the computer programs.

2-1 PROBLEM SOLVING- CASE STUDY

Problem	**Solving**
(1) There is a size for problem sentences equal to 5.	Print out the problem size
(2) A countable number of sentences or phrases.	Given a sentence by interactive means or file stream. Print out count of sentences by counting the number of characters excluding white spaces.
(3) Counting the size of selected phrases for likely occurrence of subset equality of letters is called *count*.	Given a sentence by interactive means or file stream. Tokenized each sentence into separate words. Print out the size of each word count for each sentence.

Problem	Solving
(4) A sentence with the number of letters specified is called Π.	Given a sentences by interactive means or file stream. Print out each letter with its index starting from 1 to sentence length. Take note excluding white spaces.
(5) The logical structure of arithmetic such as +, - and = should be applied.	In computing a count make sure that the sizes of each word is summed to yield the count of sentence. With that + is applied to sizes to get count.
(6) Discrete operations must include count, addition, subtraction and sizes.	The program should compute the following discrete operations count, addition, multiplication and subtraction.

Problem	Solving
(7) The sizes of selected phrases are enumerated.	Print a list of the words for a sentence and indicate the size number of each word.
(8) Proofs with the discrete operations on which the enumeration of sizes stops.	Print out discrete meta-operation summations. Print out enumeration of word sizes for a sentence.
(9) There is a possibility of summation equal to Π.	Given two sentences by interactive means or file stream. Count their total sizes and print out the total count.

Using LCProgram in solving the problems is now discussed.

2-2 LCPROGRAM DEMONSTRATION

Problem 1: Problem Size

```
INTERACTIVE INPUT:

Select an item from menu(00: Menu Items)>>>
01

OUTPUT:

Problem Size Information
========================
Problem size:5
```

Problem 2: Sentences

```
INTERACTIVE INPUT:

Select an item from menu(00: Menu Items)>>>
02

OUTPUT:

Sentence Information
============================

LCP Sentence 1:Damn it
LCP Sentence 2:Whats wrong
LCP Sentence 3:It is a combination of 46 letters
LCP Sentence 4:Akua will not marry you
```

```
LCP Sentence 5:Pokua will not marry you
```

Problem 3: Word Count in Sentence

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
04

OUTPUT:

Alpha Sizes Information
==========================

(1) Alpha size:2
(2) Alpha size:2
(3) Alpha size:7
(4) Alpha size:5
(5) Alpha size:5
```

Problem 4: Letters with indices

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
10

OUTPUT:
Alphanumerics Information
=====================================

Sentence MP(1):
D(1)  a(2)  m(3)  n(4)  i(5)  t(6)
Sentence MP(2):
W(1)  h(2)  a(3)  t(4)  s(5)  w(6)  r(7)  o(8)
n(9)  g(10)
```

```
Sentence MP(3):
I(1)  t(2)  i(3)  s(4)  a(5)  c(6)  o(7)  m(8)
b(9)  i(10)  n(11)  a(12)  t(13)
  i(14)  o(15)  n(16)  o(17)  f(18)  4(19)
6(20)  l(21)  e(22)  t(23)  t(24)  e
(25)  r(26)  s(27)
Sentence MP(4):
A(1)  k(2)  u(3)  a(4)  w(5)  i(6)  l(7)  l(8)
n(9)  o(10)  t(11)  m(12)  a(13)
  r(14)  r(15)  y(16)  y(17)  o(18)  u(19)
Sentence MP(5):
P(1)  o(2)  k(3)  u(4)  a(5)  w(6)  i(7)  l(8)
l(9)  n(10)  o(11)  t(12)  m(13)
  a(14)  r(15)  r(16)  y(17)  y(18)  o(19)
u(20)
```

Problem 5: Counting Methods

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
06

OUTPUT:
Counting Problem Information
============================

Count MP(1):6
Count MP(2):10
Count MP(3):27
Count MP(4):19
Count MP(5):20
```

Problem 6:Addition, Multiplication and Subtraction Principles

CASE 1:

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
03

OUTPUT:
MP Principle Values
============================

Addition Principle Value(MP):=82
Subtraction Principle Value(MP):=-42
Multiplication Principle Value(MP):=615600
```

CASE 2:

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
09

OUTPUT:
Principle Values for RMPSet
====================================

Addition Principle Value(RMPSet):=69
Subtraction Principle Value(RMPSet):=-29
Multiplication Principle Value(RMPSet):=319200
```

CASE 3:

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
```

```
12

OUTPUT:
Principle values for LMPSet
==================================

Addition Principle Value(LMPSet):=96
Subtraction Principle Value(LMPSet):=-42
Multiplication Principle Value(LMPSet):=8618400
```

Problem 7:Partition Sets

CASE 1:

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
05

OUTPUT:
Partition of Integers
============================

[
Sentence 1 :[4, 2]
,
Sentence 2 :[5, 5]
,
Sentence 3 :[2, 2, 1, 11, 2, 2, 7]
,
Sentence 4 :[4, 4, 3, 5, 3]
,
Sentence 5 :[5, 4, 3, 5, 3]
]
```

CASE 2:

```
INTERACTIVE INPUT:
```
Select an item from menu(00: Menu Items)>>>
17

```
OUTPUT:
```
Partition of Integers
==========
Enter the [reduce,type] of partition of integers
(Reduce Values=0 to 9)
Type Values:
(1)MP partition
(2)MPSet partition
(3)RMPSet partition
(4)LMPSet partition
(Enter values[Format:1,2])>>>2,1

######################
Partition of 10
######################
8 + 2
7 + 2 + 1
6 + 2 + 1 + 1
5 + 2 + 1 + 1 + 1

######################
Partition of 20
######################
18 + 2
17 + 2 + 1
16 + 2 + 1 + 1

15 + 2 + 1 + 1 + 1

######################
Partition of 19
######################
17 + 2
16 + 2 + 1
15 + 2 + 1 + 1
14 + 2 + 1 + 1 + 1

######################
Partition of 6
######################
4 + 2
3 + 2 + 1
2 + 2 + 1 + 1
1 + 2 + 1 + 1 + 1

######################
Partition of 27
######################
25 + 2
24 + 2 + 1
23 + 2 + 1 + 1
22 + 2 + 1 + 1 + 1

Problem 8:Meta-Sentential Operations List and Set Operations

CASE 1:

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
11

OUTPUT:
Meta-sentential Operations
==================================

[
sum(L1,s): -(sum(L5, s) , sum(L3, s))=6,
sum(L2,s): -(sum(L5, s) , sum(L2, s))=10,
sum(L3,s): -(sum(L5, s) , sum(L1, s))=14,
sum(L4,s): +|-(sum(L5, s) , sum(L4, s) , sum(L2,
s))=19,
sum(L5,s): +(sum(L3, s) , sum(L1, s))=20,
sum(L6,s): +|-(sum(L5, s) , sum(L3, s))=27]
```

CASE 2:

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
08

OUTPUT:
MP Sets Information
==============================

MP:=[6, 10, 27, 19, 20]
MPset:=[6, 10, 19, 20]
RMPset:=[6, 10, 14, 19, 20]
LMPset:=[6, 10, 14, 19, 20, 27]
```

Problem 9: Equality Principles

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
18

OUTPUT:
Equality Principles
==========================
Enter the [set1,set2] of sets

Set Values:
(1)MP partition
(2)MPSet partition
(3)RMPSet partition
(4)LMPSet partition
(Enter values[Format:1,2])>>>1,2

Count Equality Principle Value(MP:MPSet):=137
Equality Principle Value(MP:MPSet):=[12, 20, 46,
39, 20]
```

Finally, LCProgram is correct in specification and design to solve the Letter Combinatorics requirements (1) to (9).

An *r-combination* of n distinct objects is an unordered selection, or subset of r out of the n objects(letters).

Problem 10: r-combination

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

```
Select an item from menu(00: Menu Items)>>>
20


Basic Combinatorial Information (n,r)
=======================================
Enter the type of sets

Set Values:
(1)MP partition
(2)MPSet partition
(3)RMPSet partition
(4)LMPSet partition
(Enter value)>>>1

Partition Set(MP)
####################
Permutation-1 Word Arrangement:=
[6,10,27,19,20] :=[3274240320,
106342741657152000, -2292686562112045056,
6763775359274385408, 8606509298240978944]
Permutation-2 of Word Arrangements:=
[6,10,27,19,20] :=[2139095040, 2139095040,
2139095040, 2139095040, 2139095040]
Problem permutation-2 is an infinite count
process.
Combination-1 of Word Selection:=
[6,10,27,19,20] :=[4547556, 29305208790, 0, 55,
3]
Combination-2 of Word Selection:=
[6,10,27,19,20] :=[1250609096, 1356486464, 0,
1113325568, 1077936128]
```

The fundamental skills of combinatorial reasoning [1] on

letter combinatorics is simply a class of counting problems. Count Equality[1] is a principle of solution of specific classes of counting problem with no arrangement but selection with repetition.

The two main counting principles in the elements of classes of counting problem are addition and multiplication principles are solved.

- *Addition Principle[1]* states that if there are r_1 different objects in the first set, r_2 different objects in the second set, ..., and r_m different objects in the mth set, and if the different sets are disjoint, then the number of ways to select an object from one of the m sets is $r_1 + r_2 + r_3 + r_4 \ldots + r_m$.
- On the other hand, *Multiplication Principle* supposes a procedure can be broken into successive (ordered) stages with r_1 different

outcomes in the first stage, r_2 different outcomes in the second stage, …, and r_m different outcomes in the mth stage. If the number of outcomes at each stage is independent of the choices in previous stages and if the composite outcomes are all distinct, the total procedure has $r_1 \; x \; r_2 \; x \; r_3 \; x \; r_4 \; \ldots \; x \; r_m$ different composite outcomes.

A distribution problem[1] is equivalent to an arrangement or selection problem with repetition. The focus on modeling distribution problems can be broken into subcases that can be counted in terms of simple permutation and combinations. The process of distributing r identical letter objects into n different letter objects is the selection equivalence of distribution.

Distribution problem is solved.

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

Selection Equivalence of Letter Distribution[SELD]

(Corollary 1): *The distributions of identical letter objects(word) are equivalent to letter selections.*

Corollary (1) is solved.

Thus, there $C(r+n-1,r) = \frac{(r+n-1,r)!}{r!(n-1)!} \equiv SELD$

SELD is solved.

__ $C(n-r)$ ***- Combination (Corollary 2)***: *Letter combinations are a general counting method of unordered outcome.*

Corollary (2) is solved

Distributing of distinct letter objects are equivalent to arrangements but letter combinations do not have that as a generally specialized distribution problem.

__ $n \equiv^{r}$ ***- Arrangement (Corollary 1):*** Letter combinations are not a general distribution problem of distinct letter objects.

__ $n \equiv^{r}$ ***- Arrangement (Corollary 2):*** Letter combinations are not a general arrangement of ordered outcome.

Arrangement Corollaries (1) and (2) are solved.

Problem 11: Possible of Orders

```
INTERACTIVE INPUT:
Select an item from menu(00: Menu Items)>>>
22
OUTPUT:
  Possible Number of Orders(PO)
==================================
[MP: 82 x exp(4) x 2432902008176640000
, MPSet: 55 x exp(2) x 2432902008176640000
, RMPSet: 69 x exp(3) x 2432902008176640000
, LMPSet: 96 x exp(4) x 2432902008176640000
```

]

1.***(Letter Count Problem) Theorem***: *Letter combinatorics is a counting problem.*

Letter Count Problem is solved .

2.(***Letter Cut) Lemma***: *A selected phrase is by cutting a number of possible letters.*

Problem 12: Letter Cut

```
INTERACTIVE INPUT:

Select an item from menu(00: Menu Items)>>>
07

OUTPUT:

Alpha-label Information
=====================================

Alphalabel 1: [Damn, it]
Alphalabel 2: [Whats, wrong]
Alphalabel 3: [It, is, a, combination, of, 46,
letters]
Alphalabel 4: [Akua, will, not, marry, you]
Alphalabel 5: [Pokua, will, not, marry, you]
```

3.Finally, it is a counting problem and a selected phrase is by cutting a number of possible letters.

4.***(Cutting Strategy) Proposition***: *A strategy of cutting the possible matches can continue with as many comparisons as needed.*

(4) is solved by LCProgram.

5.***(Π-Sentence) Axiom***: *Π is a number of letters specified in a sentence.*

(5) is solved by LCProgram.

3-0 MARRIAGE PROBLEM (EXAMPLE)

(1) Damn it.

(2) What's wrong?

(3) It is a combination of 46 letters.

(4) Akua will not marry you.

(5) Pokua will not marry you.

Marriage Problem is combinatorially complete in count method. This means that statements from (1) to (5) are proven by carry out these provables.

---

6. *Partition of Integers (Definition): A partition of countable integer, count to be a collection of positive integers whose sum is N.*

(6) is solved by LCProgram.

7. For the Marriage Problem(MP) Example, the partition

of integers are:

Damn it. *(1)*

4 + 2 = N = 6.

What's wrong *(2)*

5 + 5 = N = 10.

The size of each of the sentences are in Appendix A is represented as LCTable.

The collection or set of a sum and the list of integers of the partition is in increasing order.

MP = { 6, 10 }, *MP* set is a set of marriage problem partitions of integers. The problem, MP 1 is equivalent to the number of integer solutions to

$2e_2 + 4e_4 = (4+2) = 6$.

The generating function for the number(6: MP 1 case), is the ways that we can choose countable integers

$(1+X^2+X^4+X^6 \ldots)\ (1+X^4+X^8+X^{12} \ldots)$.

Generating functions are to handle constraints in selection and arrangement problems.

8. Combinatorial Enumeration of Letters

- $\sum_{s} N$ is the sentential summation for a sentence.

1. $\sum_{s}^{1} N$ Is the sentential summation for sentence 1

   (From MP example): $\sum_{s}^{1} N = 6$.

2. $\sum_{s}^{2} N$ Is the sentential summation for sentence 2

   (From MP example): $\sum_{s}^{2} N = 10$.

3. $\sum_{s}^{6} N$ Is the sentential summation for sentence 3

   (From MP example): $\sum_{s}^{6} N = 27$.

4. $\sum_{s}^{4} N$ Is the sentential summation for sentence 4

(From MP example): $\sum_{s}^{4} N = 19$.

5. $\sum_{s}^{5} N$ Is the sentential summation for sentence 5

(From MP example): $\sum_{s}^{5} N = 20$.

MP={6 ,10, 27, 19, 20}

(8) is now combinatorial solved by LCProgram.

- ***Π = 46.***
- *Logical Structure*

$$L = \left(+, \sum_{s}^{4}, \sum_{s}^{3} \right)$$

- *Discrete Operation (Proof)*
  - The Equality Principle on the Marriage Problem is

$$\sum_{s}^{L} = \sum_{s}^{3} + \sum_{s}^{4}$$
$$= 19 + 27$$
$$= 46. \qquad = \Pi$$

Π stops on the enumeration of size 46 with the sentential summation for sentence 4 and sentential summation for sentence 3.

Equality Principle is solvable by LCProgram.

•*Discrete Operation (Addition)*

The Addition principle on the Marriage Problem results:

$$\sum_{s}^{L_A} = \sum_{s}^{1} + \sum_{s}^{2} + \sum_{s}^{3} + \sum_{s}^{4} + \sum_{s}^{5}$$
$$= 6 + 10 + 27 + 19 + 20$$
$$= 82$$

.

•*Discrete Operation (Multiplication)*

The Multiplication principle on the Marriage Problem results:

$$\sum_{s}^{L_M} = \sum_{s}^{1} x \sum_{s}^{2} x \sum_{s}^{3} x \sum_{s}^{4} x \sum_{s}^{5}$$
$$= 6 \; x \; 10 \; x \; 27 \; x \; 19 \; x \; 20$$
$$= 615600$$

.

•*Discrete Operation (Subtraction)*

The Subtraction principle on the Marriage Problem results:

$$\sum_{s}^{L_S} = \sum_{s}^{5} - \sum_{s}^{4} - \sum_{s}^{3} - \sum_{s}^{2} - \sum_{s}^{1}$$
$$= 20 - 19 - 27 - 10 - 6$$
$$= -42$$

.

•Discrete Meta-Operation (Addition-Subtraction)

$\sum_{s}^{L_{AS}}$ is the meta-sentential summation.

$$\sum_{s}^{L_{AS}} = \left\{ \sum_{s}^{L_1}, \sum_{s_3}^{L_2}, \sum_{s_1}^{L_3}, \sum_{s_3}^{L_4}, \sum_{s_2}^{L_5} \right\}$$

- The real principles on the Marriage Problem are based on the maximum and minimum sentential

summations: minimum=6 and maximum=20. The MP(3) is not a real marriage problem so it is not considered in MPSet={6, 10, 19, 20}. A discrete subtraction operation on the real principles gives

$$\sum_{s}^{L_{rS}} = \sum_{s}^{4} - \sum_{s}^{1}$$
$$= 20 - 6$$
$$= 14$$

The new RMPSet={ {6, 10, 19, 20} U {14} }

={ 6, 10, 14, 19, 20}.

Discrete operation on sets: addition, subtraction, multiplication and meta-operation are solvable by LCProgram.

Discrete Operations on RMPSet:

Addition Principle:

$$\sum_{s}^{L_A} = \sum_{s}^{1} + \sum_{s}^{2} + \sum_{s}^{3} + \sum_{s}^{4} + \sum_{s}^{5}$$
$$= 6 + 10 + 14 + 19 + 20$$
$$= 69$$

Multiplication Principle:

$$\sum_{s}^{L_M} = \sum_{s}^{1} x \sum_{s}^{2} x \sum_{s}^{3} x \sum_{s}^{4} x \sum_{s}^{5}$$
$$= 6 \ x \ 10 \ x \ 14 \ x \ 19 \ x \ 20$$
$$= 319200$$

Subtraction Principle:

$$\sum_{s}^{L_S} = \sum_{s}^{5} - \sum_{s}^{4} - \sum_{s}^{3} - \sum_{s}^{2} - \sum_{s}^{1}$$
$$= 20 - 19 - 14 - 10 - 6$$
$$= -29$$

4-0 SUMMARY

This is problem solving with LCProgram. This document shows the interactive input methodology used in the generating the outputs of the LC. The problem solving is methodical as 9 case studies or more and the LCProgram is used to solve each case. The case study is not just complete but correct to the specification of design. LCTable is a generative output that represents table in Appendix B, is also solvable by LCProgram at solution menu, 16. This can be verified by running the program. A pictorial depiction of the manual process of letter combinatorics are shown in Appendix A. There are about 23 explicit count methods solution provided by LCProgram with about 19 outputs in this document.

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

APPENDIX A- Size Graphics Illustration

Illustration 1: Sentence (1) Size Graphics

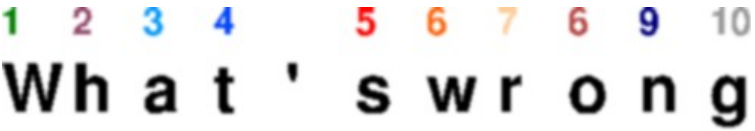
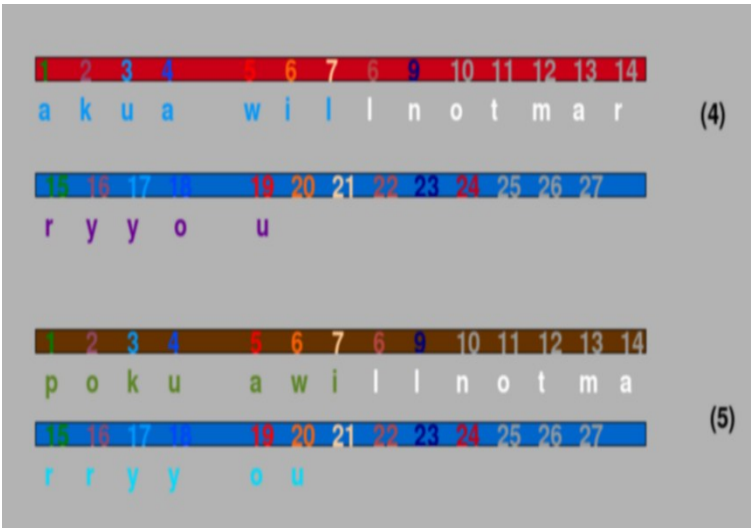


Illustration 2: Sentence (2) Size Graphics

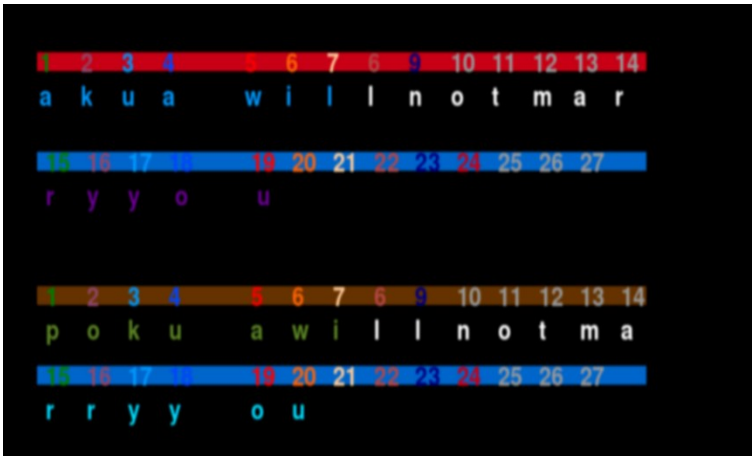
Illustration 3: Sentence (3) Size Graphics

Illustration 4: Sentence (4) Size Graphics

Illustration 5: Sentence (5) Size Graphics



Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah



Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

APPENDIX B- Table on Size, Position, Index and Words.

Index	*Position*	Word	Size
1	1	**Damn**	4
1	2	**it**	2
2	1	**What's**	5
2	2	**wrong**	5
3	1	**it**	2
3	2	**is**	2
3	3	**a**	1
3	4	**combination**	11
3	5	**of**	2
3	6	**46**	2
3	7	**letters**	7
4	1	**Akua**	4
4	2	**will**	4
4	3	**not**	3

Index	*Position*	Word	Size
4	4	**marry**	5
4	5	**you**	3
5	1	**Pokua**	5
5	2	**will**	4
5	3	**not**	3
5	4	**marry**	5
5	5	**you**	3

APPENDIX C Javajarics Output

```
Select an item from menu(00: Menu Items)>>>
00

LCProgram Menu
Explicit Count Methods
==========================

01:     Problem Size
02:     MP Sentences
03:     Principle Values (MPSet)
04:     Alpha Sizes
05:     Partition Sets
06:     Count Problems
07:     Alpha Label Tokens
08:     MP Sets
09:     Principle Values (RMPSet)
10:     Alphanumerics
11:     Meta-Operation
12:     Principle Values (LMPSet)
13:     File MP Reader
14:     Change Problem Size
15:     Change LC Problems[Interactive]
16:     LCTable View
17:     Partition of Integers
18:     Equality Principles
19:     Show Ferrers Diagrams
20:     Permutation and Combination
21:     One-to-One Correspondence
22:     Possible Number of Orders
23:     Show Pascal Triangle
Press Ctrl+C: Quit LCProgram
==========================
```

```
Select an item from menu(00: Menu Items)>>>
01


Problem Size Information
========================

Problem size:5

Select an item from menu(00: Menu Items)>>>
02


Sentence Information
============================

LCP Sentence 1:Damn it
LCP Sentence 2:Whats wrong
LCP Sentence 3:It is a combination of 46 letters
LCP Sentence 4:Akua will not marry you
LCP Sentence 5:Pokua will not marry you

Select an item from menu(00: Menu Items)>>>
03


MP Principle Values
===============================

Addition Principle Value(MP):=82
Subtraction Principle Value(MP):=-42
Multiplication Principle Value(MP):=615600

Select an item from menu(00: Menu Items)>>>
04
```

```
Alpha Sizes Information
==========================

(1) Alpha size:2
(2) Alpha size:2
(3) Alpha size:7
(4) Alpha size:5
(5) Alpha size:5

Select an item from menu(00: Menu Items)>>>
05


Partition of Integers
============================

[
Sentence 1 :[4, 2]
,
Sentence 2 :[5, 5]
,
Sentence 3 :[2, 2, 1, 11, 2, 2, 7]
,
Sentence 4 :[4, 4, 3, 5, 3]
,
Sentence 5 :[5, 4, 3, 5, 3]
]

Select an item from menu(00: Menu Items)>>>
06


Counting Problem Information
============================
```

```
Count MP(1):6
Count MP(2):10
Count MP(3):27
Count MP(4):19
Count MP(5):20

Select an item from menu(00: Menu Items)>>>
07


Alpha-label Information
=======================================

Alpha label 1: [Damn, it]
Alphalabel 2: [Whats, wrong]
Alphalabel 3: [It, is, a, combination, of, 46,
letters]
Alphalabel 4: [Akua, will, not, marry, you]
Alphalabel 5: [Pokua, will, not, marry, you]

Select an item from menu(00: Menu Items)>>>
08


MP Sets Information
==============================

MP:=[6, 10, 27, 19, 20]
MPset:=[6, 10, 19, 20]
RMPset:=[6, 10, 14, 19, 20]
LMPset:=[6, 10, 14, 19, 20, 27]

Select an item from menu(00: Menu Items)>>>
09
```

```
Principle Values for RMPSet
===================================

Addition Principle Value(RMPSet):=69
Subtraction Principle Value(RMPSet):=-29
Multiplication Principle Value(RMPSet):=319200

Select an item from menu(00: Menu Items)>>>
10

Alphanumerics Information
=====================================

Sentence MP(1):
D(1)  a(2)  m(3)  n(4)  i(5)  t(6)
Sentence MP(2):
W(1)  h(2)  a(3)  t(4)  s(5)  w(6)  r(7)  o(8)
n(9)  g(10)
Sentence MP(3):
I(1)  t(2)  i(3)  s(4)  a(5)  c(6)  o(7)  m(8)
b(9)  i(10)  n(11)  a(12)  t(13)
  i(14)  o(15)  n(16)  o(17)  f(18)  4(19)
6(20)  l(21)  e(22)  t(23)  t(24)  e
(25)  r(26)  s(27)
Sentence MP(4):
A(1)  k(2)  u(3)  a(4)  w(5)  i(6)  l(7)  l(8)
n(9)  o(10)  t(11)  m(12)  a(13)
  r(14)  r(15)  y(16)  y(17)  o(18)  u(19)
Sentence MP(5):
P(1)  o(2)  k(3)  u(4)  a(5)  w(6)  i(7)  l(8)
l(9)  n(10)  o(11)  t(12)  m(13)
  a(14)  r(15)  r(16)  y(17)  y(18)  o(19)
u(20)
```

```
Select an item from menu(00: Menu Items)>>>
11


Meta-sentential Operations
==================================

[
sum(L1,s): -(sum(L5, s) , sum(L3, s))=6,
sum(L2,s): -(sum(L5, s) , sum(L2, s))=10,
sum(L3,s): -(sum(L5, s) , sum(L1, s))=14,
sum(L4,s): +|-(sum(L5, s) , sum(L4, s) , sum(L2,
s))=19,
sum(L5,s): +(sum(L3, s) , sum(L1, s))=20,
sum(L6,s): +|-(sum(L5, s) , sum(L3, s))=27]

Select an item from menu(00: Menu Items)>>>
12


Principle values for LMPSet
================================

Addition Principle Value(LMPSet):=96
Subtraction Principle Value(LMPSet):=-42
Multiplication Principle Value(LMPSet):=8618400

Select an item from menu(00: Menu Items)>>>
13


File Sentence Stream
======================
Damn it
```

```
Whats wrong
It is a combination of 46 letters
Akua will not marry you
Pokua will not marry you
This is the end

Select an item from menu(00: Menu Items)>>>
14


Change Problem Size
=======================
Enter the size of counting problem(Format=001 to
999):
006

Select an item from menu(00: Menu Items)>>>
02


Sentence Information
==========================

LCP Sentence 1:Damn it
LCP Sentence 2:Whats wrong
LCP Sentence 3:It is a combination of 46 letters
LCP Sentence 4:Akua will not marry you
LCP Sentence 5:Pokua will not marry you
LCP Sentence 6:This is the end


Select an item from menu(00: Menu Items)>>>
18


Equality Principles
```

```
==========================
Enter the [set1,set2] of sets

Set Values:
(1)MP partition
(2)MPSet partition
(3)RMPSet partition
(4)LMPSet partition
(Enter values[Format:1,2])>>>1,2

Count Equality Principle Value(MP:MPSet):=137
Equality Principle Value(MP:MPSet):=[12, 20, 46,
39, 20]

Select an item from menu(00: Menu Items)>>>
19


Ferrers Diagram: Dot Display
============================
Enter the type of sets

Set Values:
(1)MP partition
(2)MPSet partition
(3)RMPSet partition
(4)LMPSet partition
(Enter value)>>>2
Partition Set:= MPSet

Partition Random Reducer:=2
Partition Size:4
##################
Partition of 6
##################
Partition of Integers: 4 + 2
```

```
$$Dot Partition$$

------------------------
oooo

oo
------------------------

Partition of Integers: 3 + 2 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
ooo

oo

o
------------------------

Partition of Integers: 2 + 2 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
oo

oo

o

o
------------------------

Partition of Integers: 1 + 2 + 1 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
```

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

```
o

oo

o

o

o
------------------------

##################
Partition of 10
##################
Partition of Integers: 8 + 2
$$Dot Partition$$

------------------------
oooooooo

oo
------------------------

Partition of Integers: 7 + 2 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
ooooooo

oo

o
------------------------

Partition of Integers: 6 + 2 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$
```

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

```
------------------------
oooooo

oo

o

o
------------------------

Partition of Integers: 5 + 2 + 1 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
ooooo

oo

o

o

o
------------------------

###################
Partition of 19
###################
Partition of Integers: 17 + 2
$$Dot Partition$$

------------------------
oooooooooooooooo

oo
```

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

```
------------------------

Partition of Integers: 16 + 2 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
ooooooooooooooo

oo

o
------------------------

Partition of Integers: 15 + 2 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
oooooooooooooo

oo

o

o
------------------------

Partition of Integers: 14 + 2 + 1 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
ooooooooooooo

oo

o
```

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

```
o

o
------------------------

##################
Partition of 20
##################
Partition of Integers: 18 + 2
$$Dot Partition$$

------------------------
oooooooooooooooooo

oo
------------------------

Partition of Integers: 17 + 2 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
ooooooooooooooooo

oo

o
------------------------

Partition of Integers: 16 + 2 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
oooooooooooooooo

oo
```

Letter Combinatorics Demo: Teaching Appointment by Dr. Frank Appiah

```
o

o
------------------------

Partition of Integers: 15 + 2 + 1 + 1 + 1
$$Dot Partition: Conjugate Ferrers Diagram$$

------------------------
ooooooooooooooo

oo

o

o

o
------------------------

Select an item from menu(00: Menu Items)>>>
20


Basic Combinatorial Information (n,r)
=========================================
Enter the type of sets

Set Values:
(1)MP partition
(2)MPSet partition
(3)RMPSet partition
(4)LMPSet partition
(Enter value)>>>1

Partition Set(MP)
```

```
######################
Permutation-1 Word Arrangement:=
[6,10,27,19,20] :=[3274240320,
106342741657152000, -2292686562112045056,
6763775359274385408, 8606509298240978944]
Permutation-2 of Word Arrangements:=
[6,10,27,19,20] :=[2139095040, 2139095040,
2139095040, 2139095040, 2139095040]
Problem permutation-2 is an infinite count
process.
Combination-1 of Word Selection:=
[6,10,27,19,20] :=[4547556, 29305208790, 0, 55,
3]
Combination-2 of Word Selection:=
[6,10,27,19,20] :=[1250609096, 1356486464, 0,
1113325568, 1077936128]
```

BIBLIOGRAPHY

1. *Alan Tucker(2012), Applied Combinatorics, Wiley, 6th Edition, ISBN:1118210115.*
2. *Nell Dale and Chip Weens(1994): Pascal, 4 Edition, D. C. Heath and Company.ISBN: 0-669-34219-X.*
3. *Java SDK(2019), Oracle Inc, http://www.oracle.com. ((Accessed August, 2019).*
4. *Frank Appiah(2019), Letter Combinatorics on Marriage Problem, Teaching Document, Institution Above.*
5. *Frank Appiah(2019), Letter Combinatorics on Letter Marriage Method, Teaching Document, Institution Above.*

ILLUSTRATION INDEX

INDEX